**Camera rental application**

**Project Specification:**

The project aims to develop a camera rental application that allows users to rent cameras, manage camera listings, and track wallet balances.

The application provides features for user authentication, camera listing management, camera rental, and wallet management.

***User Authentication:***

Users are required to enter a username and password for authentication. The application checks the provided credentials for successful login. If login fails user prompt with an error message displaying invalid credentials.

***Camera Listing Management:*** User can add cameras to the application's database for rental purposes. Each camera listing includes details such as camera ID, brand, model, price, and availability status. Owners can add, remove, and view their camera listings.

***Camera Rental*:**

Renters can browse the available camera listings. Renters can select a camera and proceed with the rental process. The application checks the renter's wallet balance to ensure they have sufficient funds. If the renter's balance is enough, the camera's availability status is updated, and the rental amount is deducted from the wallet balance. The renter receives a confirmation message indicating a successful rental.

***Wallet Management***: The application provides a wallet functionality to track the balance of users. Users can view their current wallet balance. Users can choose to deposit additional funds into their wallet.

**PROJECT DURATION: 8hr**

**Sprint 1: 4hr**

1. Creating Camera Authentication and main menu listing.

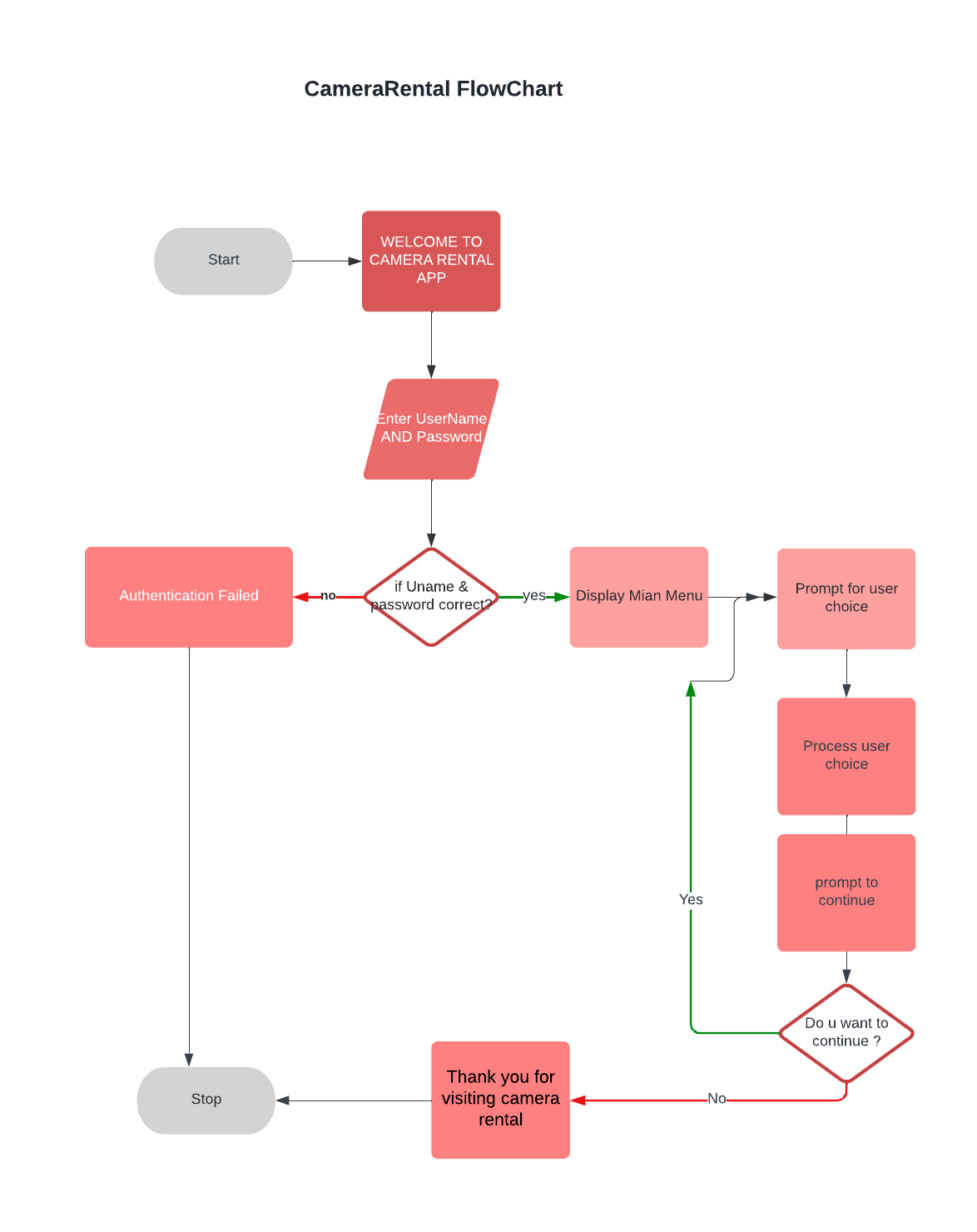
2.View and manage my camera listings.

3. Creating a class for camera data representation. Define the necessary attributes: camera ID, brand, model, price, and availability status.

4.Implement appropriate getters and setters.

**Sprint 2: 4hr**

1. Camera Rental and Wallet Management functionality.
2. Rent a camera from the available listings.
3. Deposit funds into my wallet.
4. Deduct the rental amount from the user's wallet balance if sufficient funds are available.
5. Update the camera's availability status accordingly.



**ALGORITHM:**

1.Define a class named "Camera" with properties such as ID, brand, model, price, and availability.

2.Define a class named "CameraRental" as the main class.

3.Inside the main class, create a static variable "walletBalance" to keep track of the user's wallet balance.

4.Implement the main method:

a. Print a welcome message and prompt the user for login credentials.

b. If the username and password are correct, proceed; otherwise, display an authentication failed message and exit.

c. Create a camera list by calling the "createCameraList" method.

d. Set up a loop to display the main menu options and process the user's choice until they choose to exit.

e. Prompt the user for their choice and process it accordingly:

If the choice is 1, call the "manageMyCamera" method.

If the choice is 2, call the "rentCamera" method.

If the choice is 3, call the "viewAllCameras" method.

If the choice is 4, call the "viewWalletBalance" method.

If the choice is 5, exit the program.

If the choice is invalid, display an error message and continue the loop.

f. Prompt the user if they want to continue and repeat the loop if they choose to continue.

5.Implement the "createCameraList" method to initialize and return a list of Camera objects.

6.Implement the "manageMyCamera" method:

a. Set up a loop to display the "My Camera" menu options and process the user's choice until they choose to go back.

b. Prompt the user for their choice and process it accordingly:

If the choice is 1, call the "addCamera" method.

If the choice is 2, call the "removeCamera" method.

If the choice is 3, call the "viewMyCamera" method.

If the choice is 4, return to the previous menu.

If the choice is 5, exit the program.

If the choice is invalid, display an error message and continue the loop.

c. Prompt the user if they want to add or remove a camera and repeat the loop if they choose to continue.

7.Implement the "addCamera" method to prompt the user for camera details, create a new Camera object, and add it to the camera list.

8.Implement the "removeCamera" method to prompt the user for a camera ID, find the corresponding camera in the list, and remove it if found.

9.Implement the "viewMyCamera" method to display the user's camera list with their details.

10.Implement the "rentCamera" method:

a. Display the available cameras to the user.

b. Prompt the user for the camera ID they want to rent.

c. Find the camera with the specified ID in the list.

d. If the camera is found and the user has sufficient balance, rent the camera, update the availability and wallet balance, and display a success message.

e. If the camera is not found, display an error message.

f. If the user has insufficient balance, display a transaction failed message.

11.Implement the "viewAllCameras" method to display the details of all cameras in the camera list.

Implement the "viewWalletBalance" method to display the current wallet balance and allow the user to add more money if desired.